基于微服务架构的新闻制作系统优化探索

韩 笑 李洁原
（新华社技术局，北京 100803）

摘 要：随着互联网技术的发展，各行各业的互联网化进程也在加速推进，笔者在分析微服务架构的优势和采用传统单体架构开发的新闻制作系统的不足后，利用微服务架构对新闻制作系统进行了优化和实现，提升了系统可扩展性、灵活性以及开发效率，积累了经验。

关键词：互联网技术；互联网进程；微服务架构；新闻制作系统；软件开发　　中图分类号：TP393　　文献标识码：A
文章编号：1671-0134（2021）02-062-02　　DOI：10.19483/j.cnki.11-4653/n.2021.02.016
本文著录格式：韩笑，李洁原 . 基于微服务架构的新闻制作系统优化探索 [J]. 中国传媒科技，2021（02）：62-63.

导语

随着互联网技术的发展，各行各业的互联网化进程也在加速推进，传统的软件开发理论、方法和技术已不再适应当前大型、复杂软件系统的开发。传统单体架构有开发效率低、代码维护难、扩展性差等缺点。微服务架构的出现，为大型、复杂软件系统带来了可扩展性、灵活性等多种优点，但对微服务的管理和运维提出了更高的要求。[1]

本文首先对微服务进行了介绍，然后对传统软件开发模式在大型、复杂软件系统开发中应用现状进行了分析，随后阐述了微服务划分，并利用 Spring Cloud 微服务架构对新闻制作系统进行的优化和实现。

1. 微服务简介

微服务架构是一种面向互联网应用服务的软件开发架构，主要应用于互联网应用服务的服务端软件开发，其由面向服务架构 SOA 发展而来。微服务架构提倡将单体架构应用划分成一组小的服务，服务之间互相协调、互相配合。[2]

微服务架构的优点有：服务以最小粒度被切分，每个应用都很小，边界清晰，职责单一；每个微服务都可以独立开发，对开发团队的技术栈要求更低，单个服务易于开发、理解和维护；每个微服务都可以独立部署，从而可以加快部署速度；每个微服务可以自主选择开发技术栈，可以独立更新，灵活扩展。微服务应用是分布式系统，但提高了系统开发和部署的复杂程度，存在数据一致性、服务依赖、服务监控检查等问题。

2. 现状分析

传统的单体架构（即应用程序的全部功能被一起打包作为单个单元或者应用），程序随着功能和代码量的增加，将面临维护成本增加，持续交互周期长，技术选型成本高，可伸缩性差等问题。另外大型、复杂应用一般都会面临各种各样的业务需求，传统的单体架构在初期简单易行，但会随着时间推移变得庞大而繁杂。由于提供了大量的业务功能，随着功能的升级，整个研发、发布、定位问题、扩展都会变得越来越困难。

笔者所在单位之前开发的新闻制作系统多为单体架构，大型而复杂，基于此架构后续进行敏捷开发和业务扩展都较为困难。由于部分模块没有进行有效拆分，这些模块在系统上线后，在迭代、部署和管理上都需要花费较大的精力，有的模块甚至连编译部署都需要花费一个小时之久。微服务架构相比传统单体架构所具备的优势，更适合笔者单位对媒体融合发展、智能化发展的需求。通过微服务化，当业务功能变化时，系统可以快速调整相应的服务，完成开发、测试等工作，基于云架构和平台式部署等能力，实现自动化部署，快速适应业务变化。

3. 优化设计与实现

3.1 微服务开发框架迁移

本文中尝试将笔者单位的新闻制作系统由传统服务架构改造成 Spring Cloud 微服务架构，对基础架构的迁移，主要涉及以下几个方面的变更。

服务发现方式的变更：

通过 Consul Client 向 Consul server 注册、发现、消费服务。



图 1 服务发现方式

接口定义方式的变更：

web 层调用下层服务的方式，使用基于 REST 形式的接口调用。同时集成 Spring Cloud OpenFeign，对远程 REST 调用进行自动配置和绑定，并使用断路器（Hystrix）组件防止造成雪崩效应。从而实现只对服务接口类进行改造即可达到改造目的，保证改造最少，影响最小。

新 SampleXX 服务定义的 API 接口如下：

```
@FeignClient(value= "SampleXX ",
fallback=ServiceClientHystrix.class)
public interface SampleXX {
@RequestMapping(value = "/xinhua/
getUserInfoByUserName", method = RequestMethod.GET)
String getUserInfoByUserName(@ResquestBody String
jsonStr);
}
```

服务启动方式的变更

原 Web 层服务均为部署在 Tomcat 容器，由 Tomcat 启动。现变更成 Spring Boot 独立应用启动。

3.2 服务部署架构设计

为便于未来进行分布式系统部署和管理，在虚拟化层级基础上引入了容器层。在部署架构设计中，需要对微服务依赖组件进行变更和新增，其中包括：新增应用网关服务 Spring Cloud Gateway，应用配置服务 Config Server，服务注册与发现 Consul Server/Client，断路监控面板 Hystrix Dashboard，分布式链路监控 Zipkin Server。迁移后的逻辑部署架构如下图所示。

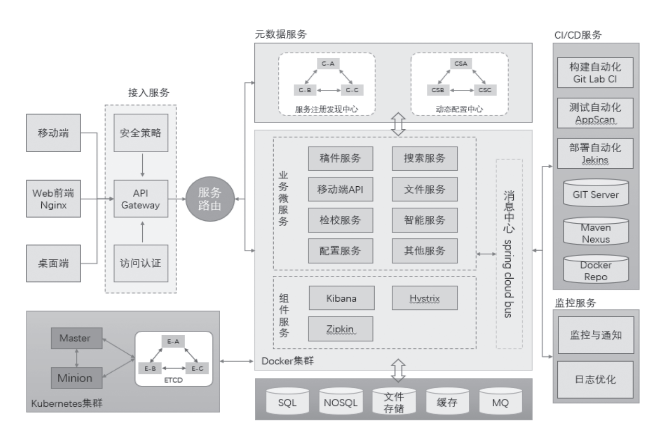


图 2 逻辑架构图

3.3 应用层微服务设计

应用层微服务按照数据服务层、数据管理层、基础功能组件层、业务服务层划分为四个层次的服务。每层为上层服务提供 API 支持，并通过业务场景层的整合和集成统一为 B/S 客户端，C/S 客户端和移动端提供服务。

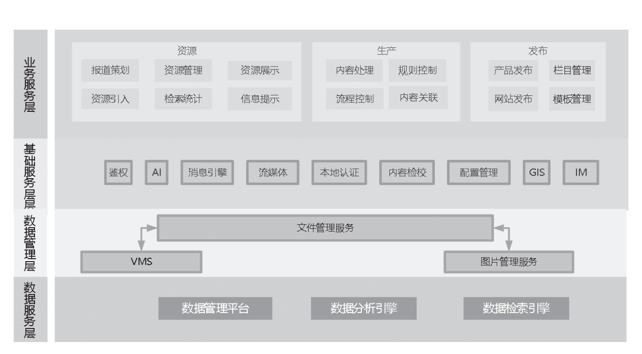


图 3 应用层服务划分图

数据服务层：集成各数据源的资源数据。

数据管理层：统一的文件管理、图片管理，一级 VMS 等基础管理服务。

基础服务层：提供鉴权服务、AI 服务、消息引擎、流媒体、本地认证授权服务、管理配置管理服务等基础服务。

业务服务层：提供资源管理、资源展示、内容处理、内容流程和规则控制、内容发布等业务服务。

以上服务将不再作为一个整体对外发布，逐个拆解封装成独立的微服务应用，每一个微服务应用都有自己的部署、资源、扩展和监控需求。

3.4 小型化部署

在对微服务进行细致拆分基础上，实现服务模块化打包、模块化持续集成、模块化部署。利用 Docker 的技术，实现基础应用及服务的跨平台快速部署和启动。使用 Kubernetes 对 Docker 容器进行统一管理，实现高负载、高可用，弹性扩展。

部署过程分成准备阶段和部署阶段。准备阶段实现按模块选择的功能。部署阶段提供模块部署副本数量及其他基础配置功能，以及多域名接入配置功能，并在配置后提供一键部署。通过上述部署过程，简化了小型化部署的操作过程，实现小型化部署的易维护性。

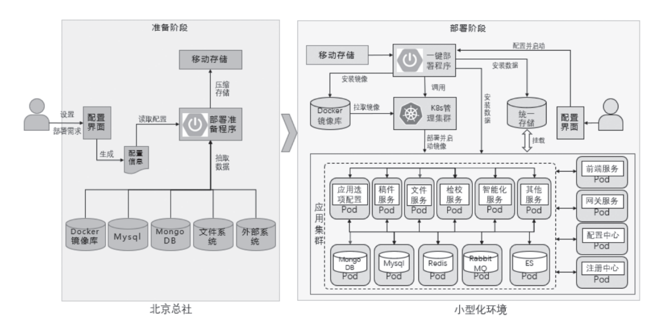


图 4 小型化部署总体流程

结语

系统建设过程中架构设计是非常重要的，需要平衡系统的可用性、可扩展性、可维护性、可靠性、高性能等需求，还要考虑到业务功能需求，所以在项目建设初期就需要明确系统建设的架构设计。为了能够支撑业务的不断扩大，需求功能的持续增加，笔者尝试对业务系统进行基于微服务架构优化设计，本文中所阐述的设计方案已经在实际工作中进行实施，在此架构上后续进行的敏捷开发工作能够快速应对业务调整及需求变化，提高了系统的适用性和扩展性，笔者所在团队将进一步总结经验，加强思考，继续探索，以期获得相关经验。媒

参考文献

[1] 吴文峻，于鑫，蒲彦均，汪群博，于笑明 . 微服务时代的复杂软件开发 [J]. 计算机科学 ,2020（12）：11–17.

[2] 赵然，朱小勇 . 微服务架构评述 [J]. 网络新媒体技术 ,2019（1）：58–59.

作者简介：韩笑（1985–），女，北京，新华社通信技术局高级工程师，研究方向：新闻制作系统建设与开发；李洁原（1983–），男，山西，新华社通信技术局高级工程师，研究方向：项目管理。

（责任编辑：张晓婧）